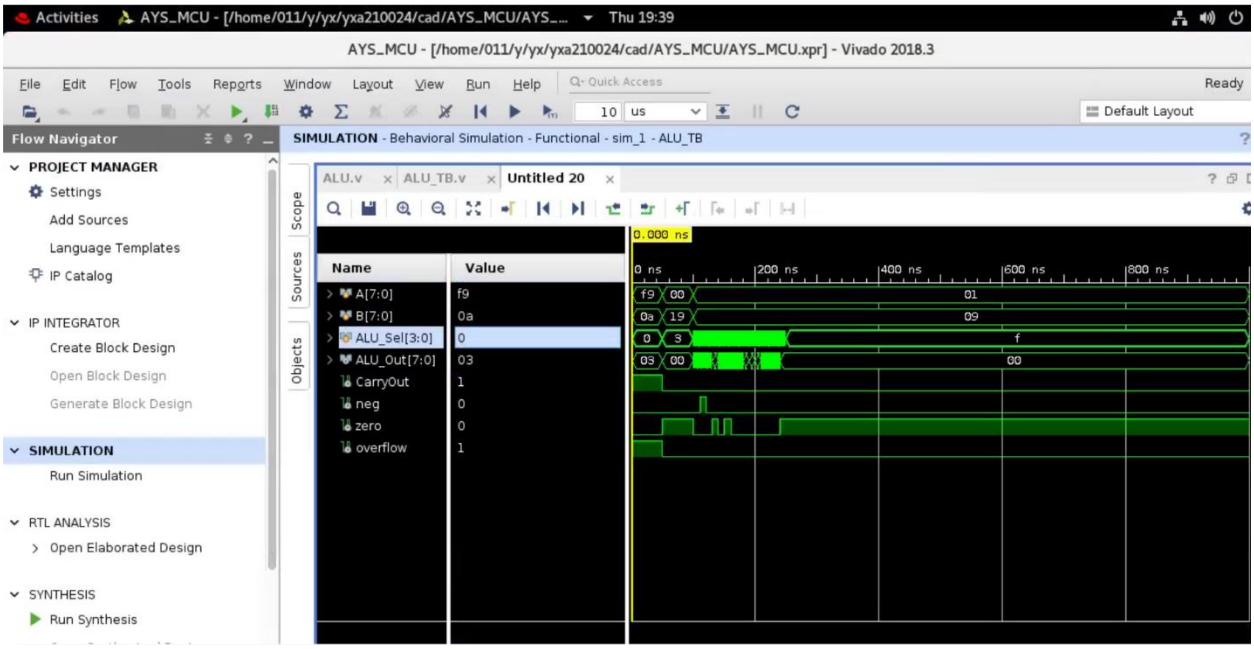
ALU SIMULATION



Cycle 1:

The first operation is to add which is 4’b0000 two numbers.

A ~ F9=249

B ~ 0a=10

When you add A+B it overflows and has a carry. Hence both the carry and overflow have become 1.

Cycle 2:

The second cycle has a division operation which is 4’b0011 of two numbers.

A ~ 00 = 0

B ~ 19 = 25

When any number is divided by 0 the output is 0 hence the zero flag is 1.

Cycle 3:

This cycle has all the operations done one by one accordingly.

A ~ 01 = 1

B ~ 09 = 9

ALU_Sel:

0

4’b0000: (Addition)

Result ~ a = 10

1

4’b0001: (Subtraction)

Result ~ -f8 = -248

Negative flag = 1

2

4’b0010: (multiplication)

Result ~ 09 = 09

3

4’b0011: (division)

Result ~ 0 = 0

zero flag = 1

4

4’b0100: (logical shift left)

A is shifted to left so 1 becomes 2

Result ~ 02 = 02

5

4’b0101: (logical shift right)

A is shifted to right so 1 becomes 0

Result ~ 0 = 0

Zero flag=1

6

4’b0110: (Rotate left)

Result ~ 02 = 2

7

4’b0111: (rotate right)

Result ~ 80 = 128

8

4’b1000: (logical and)

Result ~ 01 = 1

9

4’b1001: (logical or)

Result ~ 09 = 9

0a

4’b1010: (logical xor)

Result ~ 08 = 8

0b

4’b1011: (logical nor)

Result ~ f6 = 246

0c

4’b1100: (logical nand)

Result ~ fe = 254

0d

4’b1101: (logical xnor)

Result ~ f7 = 247

oe

4’b1110: (greater comparison)

Result ~0 = 0

Zero flag= 1

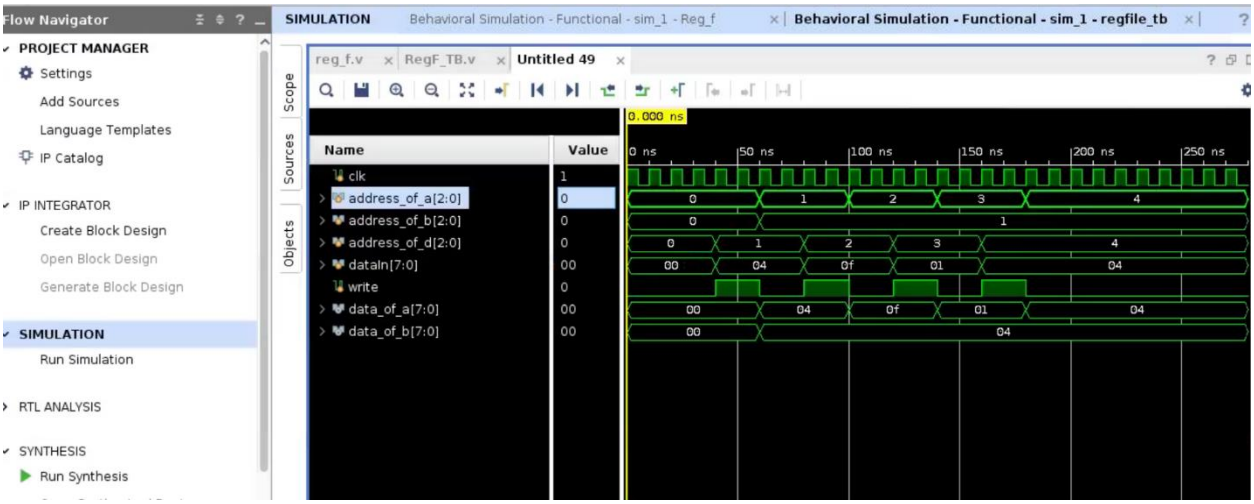
0f

4’b1111: (equal comparison)

Result ~ 0 = 0

Zero flag = 1

REGISTER FILE SIMULATION:



When write function is 0

Data_of_a reads the data from the registers at address_of_a;

Data_of_a=register[address_of_a]

Data_of_b reads the data from the registers at address_of_b;

Data_of_b=register[address_of_b]

When write function is 1

Register at address_of_d is write enabled and the data in the dataIn is written into the Register at address_of_d

Registers[address_of_d]=dataIn

At cycle 1:

All the values are initialized, and the output data is 0

At cycle 2:

Write=1;

Hence data is written into the registers at address_of_d

At cycle 3:

The data written into registers is read

At cycle 4:

The data is again written into the registers at address_of_d

At cycle 5:

Here

The newly written data is accessed by data_of_a and data_of_b retails the same value as the old one.

At cycle 6 and so on.

The data is written and read accordingly on the regular intervals at 20ns